Problem 1: Computing Systems

A wide range of computing systems are currently in production. Consider the following devices when answering the questions below: a laptop, a digital watch, a scientific calculator, a supercomputer, and a smartphone.

(a) Sketch a curve showing computational performance of all these systems as a function of their cost. Put performance on the y-axis (arbitrary units), and cost on the x-axis (dollar estimate).

(b) Similarly, show a curve that relates computational performance to system power consumption, with performance on the y-axis (arbitrary units), and power consumption on the x-axis (watt estimate). In the case of the smartphone, ignore the power consumption of the radio.
Problem 2: Logic

Consider the circuit below. All inputs \((P, Q, D_0, D_1, D_2, \text{ and } D_3)\) must be tied to 0 or 1.

\[ D_0 \quad \overline{Q} \]
\[ D_1 \quad \overline{P} \]
\[ D_2 \quad P \]
\[ D_3 \quad \overline{P} \]

(It might help to simplify this circuit, as you would with the kind of powerful diagramming tool I do not have.)

(a) What must \(D_0, D_1, D_2, D_3\) be such that \(F = P \oplus Q\)?

(b) Can any arbitrary 2 input logic function of signals \(P\) and \(Q\) be realized using the above architecture? Explain.

(a)

\(D_0 = D_3 = 0 \quad D_1 = D_2 = 1\)

(b)

Yes, the circuit is essentially equivalent to a 4-input MUX, where one of \(D_0, D_1, D_2, \text{ and } D_3\) is selected based on the values of the \(P\) and \(Q\) inputs. Each possible state of \(P\) and \(Q\) will select one of \(D_0\) through \(D_3\), and so by choosing the \(D_0\) through \(D_3\) values, we can implement any desired single-output truth table (logic function) of the \(P\) and \(Q\) inputs.

Problem 3: Logical Gates

You work for Sldgly, a start-up in San Francisco that uses the sludge found in the Bart transit system to perform logic functions. They call this device a Sludge Gate; every Sludge Gate requires a supply and ground, has input signals \(X\) and \(Y\), and produces an output signal \(Z\).
Here is the behavior of the Sludge Gate with a 5 V supply:

- $0 < Z < 5V$
- If $X$ and $Y$ have less than 2 V at the input for 5 microseconds or more, then $Z$ is greater than 4 V
- When either or both of $X$ and $Y$ have more than 3.5 V for at least 5 microseconds, then $Z$ is less than 1 V

The exact voltage at $Z$ is unpredictable and varies from Sludge Gate to Sludge Gate.

(a) Propose a simple topology using Sludge Gates to invert an input signal, $S$.

(b) For the inverter in (a) draw a very simple voltage transfer curve, making sure to label key voltages on both axes.

(c) What Boolean function of $X$ and $Y$ does the Sludge Gate implement?

(a) Either of:

(b)
(e)

NOR gate: \( Z = \overline{(X+Y)} \)
Problem 4: Structural Verilog

Write modules using structural Verilog which implement the circuit drawn below. You will need multiple `module` statements to preserve the hierarchy as drawn.
Solution:

```verilog
module majority3(
  input a,
  input b,
  input c,
  output m
);

wire and0_out, and1_out, and2_out;

// verilog primitives have no explicit port connection,
// and the output is first
and (and0_out, a, c);
and (and1_out, b, a);
and (and2_out, c, b);
or (m, and0_out, and1_out, and2_out);
endmodule

module top(
  input [2:0] x,
  input [2:0] y,
  output out
);

wire maj0_out, maj1_out;

// instantiate the modules—note the explicit port connection
majority3 maj0 (.a(x[0]), .b(x[1]), .c(x[2]), .m(maj0_out));
majority3 maj1 (.a(y[0]), .b(y[1]), .c(y[2]), .m(maj1_out));
and (out, maj0_out, maj1_out);
endmodule
```

Problem 5: Behavioral Verilog

For the logic circuit shown below, write the equivalent behavioral Verilog module which takes A and B as inputs, and gives X as output.
Solution:

module problem5(
  input A,
  input B,
  output X
);

  assign X = ~(~(~(A | ~B) | (A & B)) & (A | B));

endmodule

Note: You still receive full credit if wrote the HDL code for the simplified logic expression:

\[
X = ((A \oplus B) + A \cdot B) \cdot (A + B)
\]

\[
\Rightarrow X = \overline{A} \cdot \overline{B} + A \cdot B + \overline{A} \cdot B
\]

Or they simplified the expression complete:

\[
\overline{A}\overline{B} + \overline{A}B + A.B = \overline{A}\overline{B} + \overline{A}B + A.B + \overline{A}B
\]

\[
= \overline{A}(B + B) + B(A + \overline{A})
\]

\[
= \overline{A}(1) + B(1)
\]

\[
= \overline{A} + B
\]